**Lab Report #8**

Hanz Chua U26738740
Viraj Amarasinghe U14806439
Jian Gong U42910460

## Introduction

To further our understanding of sequential circuit design when converting stated problems into a design specification and implementation, we designed and implemented a vending machine controller circuit. This vending machine accepts nickels and dimes, dispenses candy at 25 cents, and does not return change.

## Methods and Materials

Wires
Breadboard
JK Flip-Flop ICs
D Flip FLop ICs
NAND Gate ICs
Wave Generator
Power Supply

We first derived the states according to the description listed in the diagram, showing all possible states and required conditions to move to the next state.We then derived the state table from the previously made state transition diagram, reduced the redundant states and implemented the circuit with JK flip flops. Once the circuit was constructed we set up the function generator to produce a 0.5 Hz square wave, connected all the wiring to the circuit, and tested it.

## Results

The circuit worked as planned; “adding” nickels and dimes would move the circuit to its next state, which corresponds to the current total inside the vending machine. After resolving some issues/mistakes, we concluded that the results came out as expected and contributed to the goal of the experiment; to further our understanding of sequential circuit design through designing a finite state machine.

## Conclusion

In conclusion, we produced a vending machine circuit that takes nickels and dimes and dispenses candy at 25 cents while not returning change. We ran into multiple issues, such as accidentally burning out chips and LEDs, omitting resistors, and mispositioning wiring.

**Questions**

1.

Testbench

```
module tb();
  reg [1:0]coin;
  reg clock, reset, dispense;
  wire readyToDispense;

  vendingMachine u0(coin, clock, reset, dispense, readyToDispense);

  initial begin
    clock = 0;
    dispense = 0;
    coin = 2'b00;
    $dumpfile("test.vcd");
    $dumpvars;
    forever begin
      #5 clock = ~clock;
    end
  end

  initial begin
    reset = 1;
    #2 reset = 0;
    coin = 2'b10;
    #15 coin = 2'b01;
    #25 dispense = 1;
    #5 dispense = 0;
    #40 $finish;
  end
endmodule
```

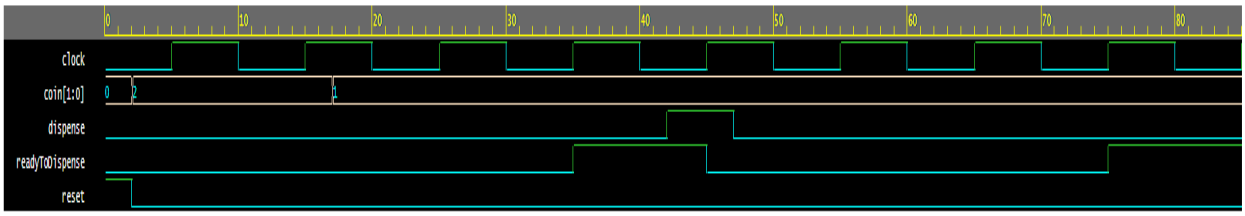
Module

```
module vendingMachine(input reg [1:0]coin, input clock, input reset,
input dispenseProduct, output reg dispense);
  reg [2:0]state;
  always @(posedge clock or posedge reset) begin
    if(reset) begin
      state = 0;
      dispense = 0;
    end
    else if(dispense && dispenseProduct) begin
```

```
  state = 0;
  dispense = 0;
end
else if(coin != 2'b11) begin
  case (state)
    (3'b000):begin
      if(coin[1]) begin
        state = 3'b001;
        dispense = 0;
      end
      else if(coin[0])begin
        state = 3'b010;
        dispense = 0;
      end
    end
    (3'b001):begin
      if(coin[1]) begin
        state = 3'b010;
        dispense = 0;
      end
      else if(coin[0])begin
        state = 3'b011;
        dispense = 0;
      end
    end
    (3'b010):begin
      if(coin[1]) begin
        state = 3'b011;
        dispense = 0;
      end
      else if(coin[0])begin
        state = 3'b100;
        dispense = 0;
      end
    end
    (3'b011):begin
      if(coin[1]) begin
        state = 3'b100;
        dispense = 0;
      end
      else if(coin[0])begin
        state = 3'b101;
        dispense = 0;
      end
    end
```

```
        (3'b100):begin
          if(coin[1]) begin
            state = 3'b101;
            dispense = 1;
          end
          else if(coin[0])begin
            state = 3'b101;
            dispense = 1;
          end
        end
        (3'b101):begin
          dispense = 1;
        end
      endcase
    end
  end
endmodule
```

## Waveform



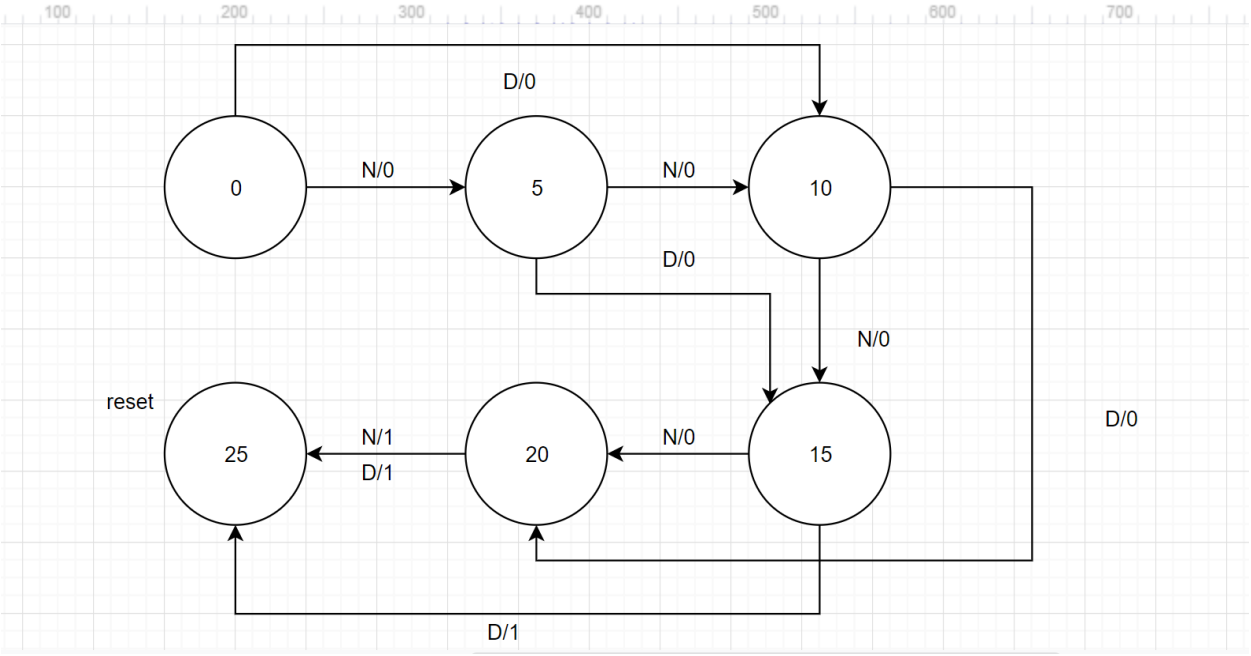
2.

Truth Table

| d | n | q2 | q1 | q0 | q2+ | q1+ | q0+ | candy | j2 | k2 | j1 | k1 | j0 | k0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | x | 0 | x |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | x | 0 | x | x | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | x | x | 0 | 0 | x |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | x | x | 0 | x | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | x | 0 | 0 | x | 0 | x |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | x | 1 | 0 | x | x | 1 |
| 0 | 0 | 1 | 1 | 0 | x | x | x | x | x | x | x | x | x | x |
| 0 | 0 | 1 | 1 | 1 | x | x | x | x | x | x | x | x | x | x |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | x | 0 | x | 1 | x |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | x | 1 | x | x | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | x | x | 0 | 1 | x |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | x | x | 1 | x | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | x | 0 | 0 | x | 1 | x |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | x | 1 | 0 | x | x | 1 |
| 0 | 1 | 1 | 1 | 0 | x | x | x | x | x | x | x | x | x | x |
| 0 | 1 | 1 | 1 | 1 | x | x | x | x | x | x | x | x | x | x |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | x | 1 | x | 0x | x |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | x | 1 | x | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | x | x | 1 | x | x |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | x | x | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | x | 1 | 0 | x | x | x |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | x | 1 | 0 | x | x | 1 |
| 1 | 0 | 1 | 1 | 0 | x | x | x | x | x | x | x | x | x | x |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 0 | 0 | 0 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 0 | 0 | 1 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 0 | 1 | 0 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 0 | 1 | 1 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 1 | 0 | 0 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 1 | 0 | 1 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 1 | 1 | 0 | x | x | x | x | x | x | x | x | x | x |
| 1 | 1 | 1 | 1 | 1 | x | x | x | x | x | x | x | x | x | x |

3. State diagram




| | | | ND | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | cents | | 00 | | 01 | | 10 | | 11 |
| | | $Q_2Q_1Q_0$ | | $Q_2^+Q_1^+Q_0^+$ | | $Q_2^+Q_1^+Q_0^+$ | | $Q_2^+Q_1^+Q_0^+$ | |
| A | 0 | 000 | A/0 | 000 | C/0 | 010 | B/0 | 001 | d |
| B | 5 | 001 | B/0 | 001 | D/0 | 011 | C/0 | 010 | d |
| C | 10 | 010 | C/0 | 010 | E/0 | 100 | D/0 | 011 | d |
| D | 15 | 011 | D/0 | 011 | F/1 | 101 | E/0 | 100 | d |
| E | 20 | 100 | E/0 | 100 | F/1 | 101 | F/1 | 101 | d |
| F | 25 | 101 | F/1 | 101 | F/1 | 101 | F/1 | 101 | d |

| $Q_2Q_1Q_0$ / ND | 000 | 001 | 011 | 010 | 100 | 101 | 111 | 110 |
|---|---|---|---|---|---|---|---|---|
| 00 | | | | | 1 | 1 | d | d |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 01 | | | 1 | 1 | 1 | 1 | d | d |
| 11 | d | d | d | d | d | d | d | d |
| 10 | | | | 1 | 1 | 1 | d | d |

$Q_2^+=Q_2+Q_2'Q_1D+Q_0'ND'$

| $Q_2Q_1Q_0$ / ND | 000 | 001 | 011 | 010 | 100 | 101 | 111 | 110 |
|---|---|---|---|---|---|---|---|---|
| 00 | | | 1 | 1 | | | d | d |
| 01 | 1 | 1 | | | | | d | d |
| 11 | d | d | d | d | d | d | d | d |
| 10 | | 1 | 1 | | | | d | d |

$Q_1^+=Q_2'Q_1N'D'+Q_2'Q_1'D+Q_2'Q_0N$

| $Q_2Q_1Q_0$ / ND | 000 | 001 | 011 | 010 | 100 | 101 | 111 | 110 |
|---|---|---|---|---|---|---|---|---|
| 00 | | 1 | | 1 | | 1 | d | d |
| 01 | | 1 | | 1 | 1 | 1 | d | d |
| 11 | d | d | d | d | d | d | d | d |
| 10 | 1 | | 1 | | 1 | 1 | d | d |

$Q_0^+=Q_2Q_0+Q_0'D+Q_2Q_1'N+Q_2'Q_1Q_0'N'+Q_2'Q_1Q_0N+Q_2'Q_1'Q_0N'+Q_2'Q_1'Q_0'N$

4. We implement the circuit with JK flip flops due to the lack of D flip flops, but simulated with D flip flops for simplicity.

5. I made the assumption that because we do not return change and can go over 25 cents, we can make the 20 cent state's next state the "dispense" state. I also made the assumption that the circuit resets once we exceed 25 cents.